Setting Up Continue.dev with Ollama for Local LLMs in VSCode

danielkliewer.com/2024/12/19/continue.dev-ollama

Prerequisites

1. **VSCode + Continue.dev**: Ensure you have Visual Studio Code installed and the Continue.dev extension installed.
2. **Ollama**: Install Ollama, a local LLM runner that can host various models. Make sure Ollama is running and that you know the port it's listening on (default: `11434`).

Step-by-Step Instructions

1. Start Ollama

- Run `ollama server` or ensure Ollama is already running in the background. By default, Ollama exposes its API at `http://localhost:11434`.
- You can verify this by navigating to `http://localhost:11434/version` in your browser or using `curl http://localhost:11434/version`.

2. List Available Models in Ollama

To know which models Ollama currently manages, run:

```
ollama ls
```

This will output something like:

```
qwen2.5-coder:3b
llama-2-7b
mistral-7b
...
```

Each line shows a model identifier you can use in the Continue.dev configuration. Models managed by Ollama often follow the format: `modelName:variantOrSize`, for example `qwen2.5-coder:3b`.

3. Configuring Continue.dev's `config.json`

Continue.dev reads its model configuration from a JSON file which you can typically find in your VSCode settings directory for Continue. The configuration might look like this (adjust the path as necessary):

- On Linux/MacOS, a common location might be `~/.continue/config.json`.
- On Windows, it might be in your user directory under a `.continue` folder. If you're unsure, refer to the Continue.dev documentation or run the `Continue: Open Config` command from the VSCode command palette.

Inside the `config.json`, you'll have a `models` array. To integrate an Ollama model, you need to add an entry for it. A minimal example looks like this:

```
{
  "models": [
    {
      "title": "Qwen 2.5 Coder 3b",
      "model": "qwen2.5-coder:3b",
      "provider": "ollama",
      "apiBase/v1": "http://localhost:11434/api/generate"
    }
  ]
}
```

Key Points:

- **`title`**: A human-friendly name for your model as it will appear in Continue's model selection.
- **`model`**: The exact name of the model as listed by `ollama ls`. This includes any tags like `:3b` or `:7b`.
- **`provider`**: Set this to `"ollama"` so Continue knows to route prompts to the Ollama backend.
- **`apiBase/v1`**: This must point to Ollama's API endpoint for generating responses. By default, Ollama listens on `http://localhost:11434/api/generate`. Make sure this is included exactly as shown.

You can add as many models as you like by including multiple objects in the `models` array, for example:

```
{
  "models": [
    {
      "title": "Qwen 2.5 Coder 3b",
      "model": "qwen2.5-coder:3b",
      "provider": "ollama",
      "apiBase/v1": "http://localhost:11434/api/generate"
    },
    {
      "title": "Llama 2 7B",
      "model": "llama-2-7b",
      "provider": "ollama",
      "apiBase/v1": "http://localhost:11434/api/generate"
    }
  ]
}
```

4. Loading Models into Ollama

Option A: Pulling Models from a Remote Source

If a model is hosted in a repository or by Ollama itself, you can pull it directly:

```
ollama pull qwen2.5-coder:3b
```

This downloads the model files into Ollama's directory. Once pulled, you can list it with `ollama ls` and add it to `config.json`.

Option B: Loading a Local GGUF Model

If you have a GGUF model file on your local machine (for example, `my-model.gguf`), you can integrate it with Ollama by creating a custom model YAML file that tells Ollama how to load it. Ollama's documentation details this process, but it typically looks like:

1. Create a model YAML file (e.g. `my-model.yaml`) in your Ollama models directory (commonly `~/.ollama/models/`):

   ```
   name: my-local-model
   model: /path/to/my-model.gguf
   ```

2. Once you have the YAML file in place, run:

   ```
   ollama import my-local-model.yaml
   ```

   This makes Ollama aware of the model.

3. After importing, you can verify it's recognized:

   ```
   ollama ls
   ```

   You should see `my-local-model` listed.

4. Add the model to Continue's `config.json`:

   ```
   {
     "models": [
       {
         "title": "My Local Model",
         "model": "my-local-model",
         "provider": "ollama",
         "apiBase/v1": "http://localhost:11434/api/generate"
       }
     ]
   }
   ```

5. Using the Models in VSCode with Continue.dev

- After editing `config.json`, restart Visual Studio Code or run `Continue: Reload` command from the command palette if available.
- Open the Continue.dev panel (usually on the sidebar or by using the `Continue: Open` command).
- Select the desired model from the model dropdown at the top of the Continue panel.
- Start interacting with the model. Your queries and code completions should now route through Ollama's locally hosted model.

6. Troubleshooting

- **Connection Issues**: If Continue can't reach Ollama, verify the `apiBase/v1` URL and port. The default should be `http://localhost:11434/api/generate` unless you changed Ollama's default port.
- **Missing Models**: If a model doesn't show up, verify it's listed by `ollama ls` and that you spelled it correctly in `config.json`.
- **File Permissions**: On some systems, ensure you have the correct file permissions for the `.continue` directory and the Ollama model directories.

Summary:
To integrate Ollama with Continue.dev in VSCode, you need to edit your `config.json` to include a model entry pointing to Ollama's `apiBase/v1` endpoint and referencing the model's name exactly as Ollama recognizes it. You can load models by pulling them with `ollama pull` or importing a local GGUF file via a model YAML. After configuration, you can switch between any models you've added directly from Continue.dev's interface in VSCode.